Software Engineering Institute

T-Check in Technologies for Interoperability: Business Process Management in a Web Services Context

Fabian Hueppi
Lutz Wrage
Grace A. Lewis

September 2008

TECHNICAL NOTE
CMU/SEI-2008-TN-005

Integration of Software-Intensive Systems (ISIS) Initiative

Carnegie Mellon

This report was prepared for the

SEI Administrative Agent
ESC/XPK
5 Eglin Street
Hanscom AFB, MA 01731-2100

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

For information about purchasing paper copies of SEI reports, please visit the publications portion of our Web site (http://www.sei.cmu.edu/publications/pubweb.html).

Table of Contents

List of Figures

List of Tables

Abstract

In Business Process Management (BPM), many technologies are available to describe, analyze, execute, and monitor business processes. Composition languages are one type of BPM technology. Through the use of composition languages, business processes that are implemented through software and available as web services can be combined into new processes. The most popular language in this field is the Business Process Execution Language (BPEL). BPEL allows a user to declaratively combine existing services within and outside an organization to implement a full business process. This technical note presents the results of applying the T-Check approach in an initial investigation of BPEL and related technologies for the implementation of BPM. This approach involves (1) formulating hypotheses about the technology and (2) examining these hypotheses against specific criteria through hands-on experimentation. The outcome of this two-stage approach is that the hypotheses are either fully or partially sustained or refuted. In this report, three hypotheses are examined: (1) business process descriptions can be exchanged between different design tools and runtime engines; (2) the development effort for integration is reduced through the use of a BPM tool; and (3) business processes can be changed dynamically at runtime. From the T-Check investigation, the first two hypotheses are partially sustained and the last hypothesis is fully sustained.

1 Introduction

Business Process Management (BPM) is an emerging field of knowledge and research at the intersection between management and information technology. BPM encompasses methods, techniques, and tools to design, enact, control, and analyze operational business processes involving humans, organizations, applications, documents, and other sources of information [Wikimedia 2007]. The focus of this report is the use of the T-Check approach for the examination of some technologies and standards that are currently being used to implement BPM in a web services context.

A T-Check[SM] investigation is a simple and cost-efficient way to understand what a technology can and cannot do in a specific context [Lewis 2005]. Specifically, this T-Check investigation focuses on finding initial answers to the following questions:

1. What standards are used to enable BPM with Web services?
2. Can artifacts adhering to these standards be exchanged easily between different BPM tools?
3. Can the development effort for BPM solutions be reduced by using tools that support these standards?

In the rest of this section, we briefly introduce the relevant technologies. In section 2, we describe BPM concepts and standards. Section 3 presents how we evaluate BPM technology; section 4 describes our sample implementation; and section 5 presents the evaluation results. We end the document with an outlook on future work and conclusions about the investigation in the final two sections.

1.1 WEB SERVICES

A web service has been defined by the World Wide Web consortium (W3C) as follows [W3C 2004]:

> *… a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.*

Web services offer one approach to implementing service-oriented architecture (SOA), where the following conditions apply:

- Service interfaces are described using Web Services Description Language (WSDL) [W3C 2005a].
- Message payload is transmitted using Simple Object Access Protocol (SOAP) over HTTP (Hypertext Transfer Protocol) [W3C 2003].
- Universal Description Discovery and Integration (UDDI) is used for service discovery [OASIS 2005]. Its use is optional.

Other combinations of technologies can be used to implement SOA, but using web services is by far the most common approach. For this reason, the acronym SOA is often used to imply the use

of web services as the implementation technology. For a T-Check investigation of web services see *Model Problems*[1] *in Technologies for Interoperability: Web Services* [Lewis 2006].

1.2 BUSINESS PROCESSES

The term *business process* has been used in industry for a long time. Broadly defined, a business process describes a series of steps that need to be performed in order to provide a good or a service; it also delineates the resources required. Some common examples of business processes in industry are accounts receivable, credit approval, inventory management, and shipping. The effectiveness and efficiency with which these processes are executed are critical to the success of any organization. With increased competition, organizations have raised their attention to these processes and tried to optimize or completely reengineer them. For many companies, the first step is to capture and codify existing business processes. This provides the foundation for an analysis and possible later improvement of the processes based on properties such as throughput rate, flow time, or inventory. The most common notations for modeling or capturing business processes are flow charts, the Unified Modeling Language (UML), and more recently the Business Process Modeling Notation (BPMN) [OMG 2006, OMG 2007]. There are also many tools available in the market for the modeling and analysis of business processes.

1.3 BUSINESS PROCESS MANAGEMENT

BPM is relevant not only to managing an organization's operations but also IT, because the majority of activities in a business are supported by software systems. These automated activities are the focus of our T-Check investigation. BPM includes the analysis, modeling, execution, and monitoring of a business process; it also involves the coordination among multiple business processes. In this report, we focus on the modeling of business processes for execution and subsequent coordination of these processes. We use *business process* to refer to automated activities and *workflow* when human activities are included.

1.4 BUSINESS PROCESS MANAGEMENT AND WEB SERVICES

During the modeling of a business process, the steps that need to be performed to achieve the goal of the process are identified. The modeling often starts from an existing workflow with the intent to automate as many steps as possible. One activity during modeling is to identify how each step can be implemented and whether a technology is available to do so. The necessary functionality may already exist in legacy systems, or it may be developed from scratch. Multiple legacy platforms typically contain existing functionality; new functionality is often developed using different and more modern technologies. Thus, the composition of these process steps into a new business process can become an integration challenge.

The Web service technology stack provides the ability to access and integrate business functionality that runs on multiple heterogeneous platforms, as long as the functionality is exposed as Web services [W3C 2005b]. Meeting that requirement may call for the development of wrapper services around existing code to implement a BPM scenario [Leymann 2002]. Web service technologies also allow the newly composed business processes to be exposed as Web services, hiding the complexity of the underlying process implementation.

[1] The T-Check approach was called the model problem approach previously and is referred to as such in other Carnegie Mellon® Software Engineering Institute (SEI) technical notes and reports. (Carnegie Mellon is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.)

Splitting a business process into steps that are implemented as independent Web services allows an organization to be agile in changing its business processes. These services are loosely coupled, making it relatively easy to add new steps or remove obsolete ones from the process. Ideally, new business processes can be implemented by reusing existing Web services as steps of the process, and the implementation of one step can be replaced with a newer Web service—even a service provided by a third party—without having to modify and retest the whole process.

2 BPM Specifications for Web Services

The basic Web service standards, Simple Object Access Protocol (SOAP) and Web Service Definition Language (WSDL), specify service interfaces and the messages that can be exchanged with a service. BPM specifications for Web services are at a higher level and allow specifying the order of service invocations and relationships between messages. Entire processes can be defined instead of a single interaction with a service. In this section, we explain common BPM terminology and standards.

2.1 COMMON TERMS USED IN A BPM CONTEXT

Over the past years, many terms and standards have evolved in BRM. These terms are often confused or used interchangeably. This section provides a quick overview and defines how the terms are used in this report.

Orchestration

An orchestration

- specifies internal actions as well as external interactions
- controls the complete process
- can be executed on an orchestration engine
- is not visible to participating services

In a business process context, the term *orchestration* refers to the composition of processes from existing services. An orchestration defines external interactions as well as internal actions. External interactions include invoking other services and receiving incoming messages; internal actions include data transformations and calculations.

An orchestration controls the composed process, which can be executed by an orchestration engine. This engine is responsible for instantiating processes when client requests arrive, performing necessary data transformations, and calling external services. Orchestrations also interpret incoming replies for the calling process. To a participating service, an orchestration is just another client. Orchestrations are themselves exposed as services that can participate in other orchestrations.

Figure 1 shows a simple view of an orchestration. In this illustration, a client calls the orchestration service representing the business process. Subsequently, the orchestration calls the participating services (Services 1-3) it needs in order to achieve the goal of the process.

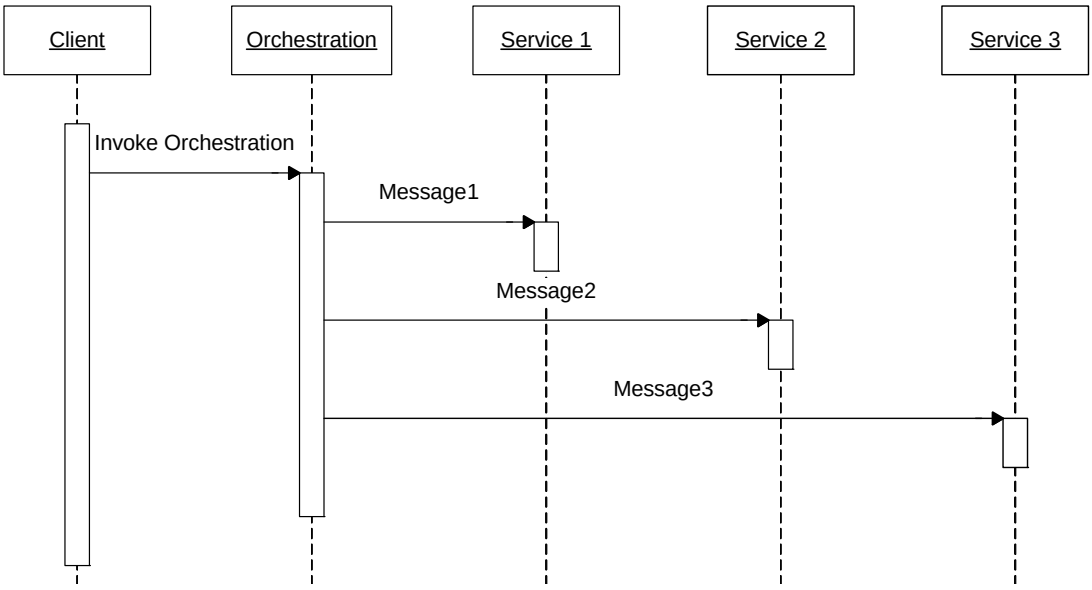

Figure 1: Conceptual View of an Orchestration (UML Sequence Diagram)

Choreography

A choreography

- specifies the externally observable behavior of a collection of collaborating services
- is not executable
- has no single controlling process

A choreography describes service interactions from the perspective of an external observer, seeing only what is happening at the services' interfaces; it does not describe what is happening inside a participating service. In essence, it describes a protocol of interaction among services collaborating to provide a service. This protocol specifies aspects such as the order of messages exchanged, types of messages, and timing constraints. Although not executable, a choreography is useful at both design time and runtime of a service-based software system. At design time, a choreography can be used as a process specification from which service interfaces can be derived. At runtime, a choreography can be used to monitor service interactions. In a choreography, participating services must have knowledge about the overall process they are performing; there is no controlling process. In particular, if the choreography describes an interaction of a service S_1 with another service S_2, this interaction must be initiated by service S_1.[2] In contrast, service interactions in an orchestration can be initiated by the orchestration engine.

Steve Ross-Talbot, co-chair of the W3C Web Services Choreography Working Group, describes the difference between orchestration and choreography as follows: "Orchestration gets realized as an executable, so you can identify the conductor in the pit and realize that he's as much an actor as the violinist. Choreography, on the other hand, is a description. The choreographer writes the de-

[2] The services participating in a choreography may be owned by multiple organizations. In such a situation, all participating organizations must agree on the choreography. No single organization can own and control the choreography.

scriptions down and gives it to the dancers and works with them to make sure they learn their parts, but he's not there as an ‘executable actor’ when it's happening” [SearchSOA 2005].

Figure 2 shows a simple view of a choreography. In this illustration, a client interacts with a service participating in a choreography (Service 1). This service then invokes another service (Service 2), which then interacts with other services (Services 3 and 4). All of these interactions are described in the choreography. When they receive a message, the participating services in the business process know the messages to send and the services to send them to. Figure 2 also shows that there is no central process has control over all service interactions, as opposed to Figure 1 where the orchestration engine is in charge of all service activities.

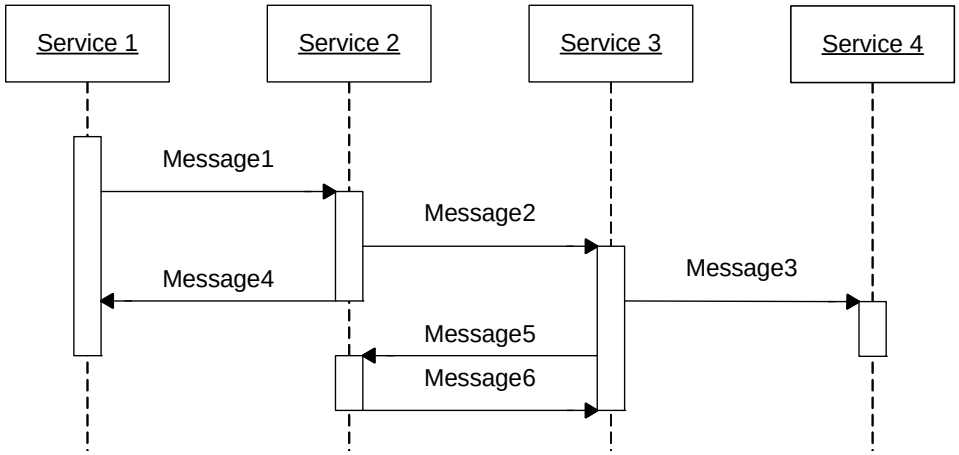

Figure 2: Conceptual View of a Choreography (UML Sequence Diagram)

Composition

The term *composition* refers to the combining of existing processes to form a new process. Orchestration and choreography are types of service composition.

Coordination

The term *coordination* is most often used to refer to the WS-Coordination standards that allow the management of context information between interacting services [OASIS 2007a]. There are two related standards, WS-Transaction and WS-BusinessActivity [OASIS 2007b, OASIS 2007c]. WS-Transaction allows for Web services to implement transactions using a two-phase commit protocol; WS-BusinessActivity deals with long-running transactions. Some of the concepts from WS-BusinessActivity have been incorporated in the long-running transactions of the Web Services Business Process Execution Language (WS-BPEL) [OASIS 2007d].

Workflow

The term *workflow* in a BPM context generally refers to a process that involves human interaction. A typical application of a workflow would be to specify the order in which dialogs or screens are displayed within a computer program. A workflow would also allow making the order dependent on user inputs. These human interactions are missing in WS-BPEL.

2.2 COMMON STANDARDS IN THE AREA OF BPM

Before describing the T-Check approach, we provide a quick overview of some important standards in the area of BPM.

2.2.1 Business Process Execution Language (BPEL)

BPEL is a declarative language for the orchestration of business processes. BPEL originated in Microsoft's XLANG (an extension of WSDL) and IBM's Web Services Flow Language (WSFL). The most current version is WS-BPEL 2.0 [OASIS 2007d].

A BPEL process makes use of Web services to fulfill its purpose. A BPEL process consists of a number of activities that are combined to implement the intended process logic. BPEL process definitions are written in XML, where activities are represented as XML elements. There are two classes of activities: basic and structured. Basic activities are individual steps. An example of a basic activity is <receive>, which pauses the process until an expected message arrives. Table 1 shows a list of basic BPEL activities. A basic activity that is an important element of an orchestration is the <assign> activity, which extracts data from one message and translates it into the format required by a subsequent activity. BPEL uses XPath [W3C 1999] as the language to reference these message parts.

Table 1: Basic BPEL Activities

Basic Activity Name	Action
<invoke>	Call a Web service. The attributes specify the input and (if available) the output message.
<receive>	Wait for the arrival of a message. Each BPEL process has at least one <receive> activity that waits for a request from a client.
<reply>	Send back a reply message if the process execution is synchronous
<assign>	Update variable values or endpoint references in partner links.
<throw>, <rethrow>	Signal a fault. If a fault handler is defined for the raised fault, this handler is executed.
<exit>	Terminate a process immediately. No fault handling will be executed
<wait>	Wait for a specified time before resuming the process
<empty>	Do nothing. This activity can, for example, be useful in a fault handler, when the process should just ignore the error.

Structured activities stipulate the order in which a set of activities is to be executed. An example of a structured activity is <forEach>. The <forEach> activity is the root tag for a loop that executes other activities repeatedly. A BPEL process can execute activities in order or in parallel. Because a BPEL process is a Web service, the orchestration can be referenced and accessed as such, even from other BPEL processes. An example of a BPEL process can be found in Appendix A. This process was implemented as part of this T-Check and is described in detail in Section 4.1. Table 2 lists the BPEL structured activities.

Table 2: Structured BPEL Activities

Structured Activity Name	Description
<sequence>	Contains one or more activities that are performed sequentially
<if>	Provides conditional behavior
<while>, <repeatUntil>	Executes a contained activity repeatedly based on a condition
<pick>	Waits for the occurrence of exactly one event from a set of events, then executes the activity associated with that event
<flow>	Provides concurrency and synchronization
<forEach>	Executes a contained activity a certain number of times based on a counter

BPEL supports long-running transactions, where the complete execution of a process may take an extended period of time, such as weeks. This makes it infeasible to use a mechanism such as two-phase commit to implement these transactions. Instead, BPEL provides a mechanism called compensation. A *compensation* process is executed when a successfully performed activity needs to be undone at a later point in time. The canonical example scenario for compensation is the travel agency orchestration. To book a trip, a travel agent service needs to contact three external services: flight reservation, hotel reservation, and rental car reservation. Assume that after the travel agent service has successfully booked a flight and a hotel for the trip, the whole itinerary needs to be canceled because there is no rental car available. Since the two reservations are already completed successfully, the travel agent service now needs to execute compensating activities, which in this case send cancellation requests to the flight and hotel reservation services.

Another important concept in BPEL is *correlation*. Since many instances of the same business process can run on a single execution environment, outgoing and incoming messages need to carry information to identify process instances, such that the engine can deliver each message to the correct process instance.

A business process designer combines BPEL activities to achieve an intended outcome. Although this work is usually done using a graphical editing tool, it is important to note that BPEL does not specify any graphical format to express an orchestration. A BPEL process is specified in the form of an XML document.

The main limitations of BPEL are that (1) it does not include ways to express human interactions and (2) all services participating in the orchestration must be Web services. The first issue is being addressed by a BPEL extension called BPEL4People that was initiated by IBM and SAP [IBM 2007]. The restriction that only Web services can be used for interaction is a problem for organizations where most of the business logic is contained in legacy applications. Using BPEL in such an environment requires adding a Web service wrapper for each of these applications, which is not always architecturally wise.

BPEL is available in a non-executable form called abstract BPEL. An abstract BPEL process defines only the external, visible characteristics of a process, such as service interfaces, messages, and constraints on the order in which messages may arrive at the service. Basically, an abstract BPEL process is the same as an executable one, except that all decisions and transformations that happen within an orchestration are removed. The difference between a choreography and an abstract BPEL process is that the latter specifies only the external behavior of one service, whereas the former talks about the interaction of multiple services.

2.2.2 Business Process Modeling Notation (BPMN)

BPMN is a graphical standard for the definition of business processes; it is under supervision of Object Management Group [OMG 2007]. The goal of this notation is to define a universal graphical language (much like UML) to describe business processes that can be understood by business analysts and technical people. Even though BPMN does not specify any underlying implementation, the specification has an appendix that explains how BPM constructs could be expressed in BPEL. Unfortunately, this explanation does not cover the entire range of BPMN features. Figure 3 (created with Intalio BPMS Designer [Intalio 2008]) shows a BPMN view of the Order Processing business process that is used in the scenario described in Section 3.1. The small rectangles are tasks. The big rectangle is a loop that iterates over the tasks inside the loop. There are also two external processes that interact with the main process via messages.

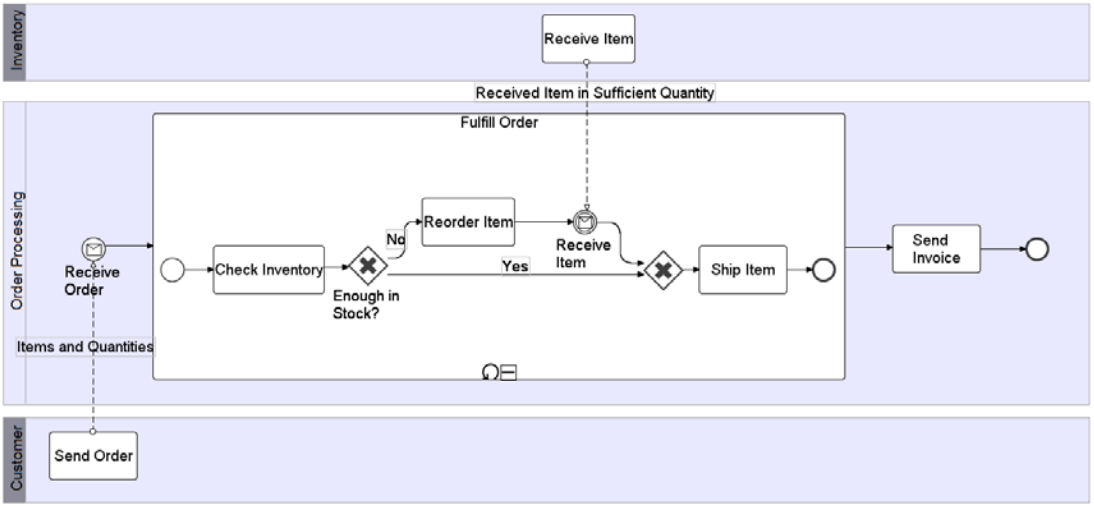

Figure 3: BPMN Diagram of the Order Processing Business Process

2.2.3 Web Service Choreography Description Language (WS-CDL)

WS-CDL is a Candidate Recommendation to the Worldwide Web Consortium (W3C) as a standard to describe choreographies [W3C2005c]. A WS-CDL process is not executable because it describes the relationship between multiple participating services. Participating services can be Web-service-implemented in any way, including a process definition language such as BPEL. At the time of writing, tool support for WS-CDL in industry is very limited.

2.2.4 XML Process Definition Language (XPDL)

XPDL is a standard to express workflows [WfMC 2005]. XPDL-compliant artifacts can be exchanged between workflow design tools and execution runtimes from different vendors. XPDL has the capability of storing graphical information about a process diagram, such as the X and Y coordinates of diagram elements.

Beginning with version 2.0, XPDL also defines how the graphical representation of the BPMN can be stored in the XPDL XML-file format. So, although BPMN does not prescribe a standard for its file format, vendors can now use XPDL to make their diagrams interchangeable.

3 Using the T-Check Approach

The T-Check approach is a technique for evaluating technologies. This approach involves (1) formulating hypotheses about the technology and (2) examining these hypotheses against specific criteria through hands-on experimentation. The outcome of this two-stage approach is that the hypotheses are either sustained (fully or partially) or refuted. The T-Check approach has the advantage of producing very efficient and representative experiments that not only evaluate technologies in the context of their intended use but also generate hands-on competence with the technologies [Wallnau 2001]. A graphical representation of the four-phase T-Check process is shown in Figure 4.

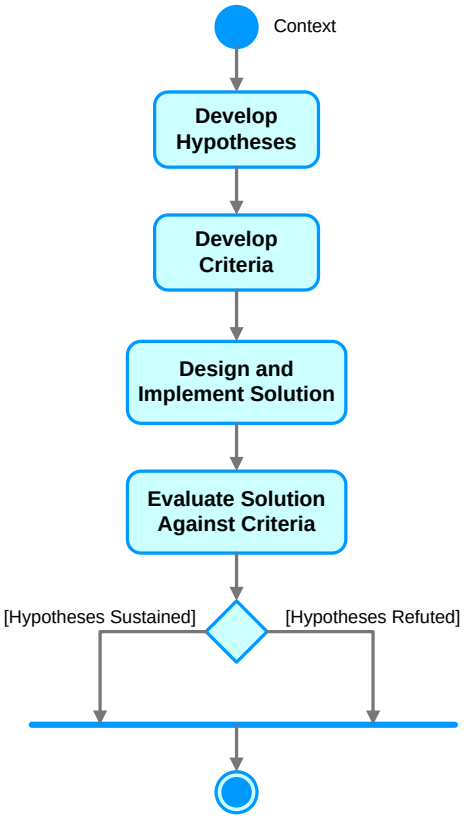

Figure 4: T-Check Process for Technology Evaluation

The T-Check approach is part of a larger process for context-based technology evaluation. In this larger process, the context for the T-Check is established and the expectations from the technology are captured [Lewis 2005].

3.1 T-CHECK CONTEXT

The context for this T-Check investigation is an organization that sells a diverse set of products. To fulfill customer orders, the organization uses an application developed in-house. This application is integrated with the accounting system, so that a bill can be sent to the customer once an order is fulfilled. Other systems, such as the inventory and shipment application, are not yet integrated. Suppliers are contacted via phone or e-mail when products need to be reordered.

The organization has suppliers and customers that wish to do business over the Internet using Web services. The organization sees this as a chance to fully integrate its existing applications. As a first step, Web service wrappers were created around some of the legacy systems. The organization is now contemplating whether the new integrated processes should be based on the BPEL standard. One reason for considering this standard is that it seems very easy to build processes with BPEL, which would be an advantage given the organization's small development staff. Also, because it seems likely that the organization will undergo a merger in the future, it wants to make sure that business processes can be easily exchanged with new partners or deployed on a different platform.

The organization wants to answer the following questions:

1. Can different BPM tools exchange business process definitions?
2. Can BPM tools hide the complexity of process orchestration, or does IT staff have to be trained in BPM concepts?
3. How easy will it be to change deployed business processes in case of a merger?

3.2 HYPOTHESES FOR THIS T-CHECK

For BPM in a Web services context, we defined the following initial hypotheses, based on claims found in experience reports and on vendor Web sites:

1. Business process descriptions can be exchanged between different design tools and runtime engines.
2. The development effort for integration is reduced through the use of a BPM tool.
3. Business processes can be changed dynamically at runtime.

3.3 CRITERIA FOR THE HYPOTHESES

Table 3 shows the evaluation criteria for the above hypotheses.

Table 3: Evaluation Criteria

Hypothesis	Criteria
Business process descriptions can be exchanged between different design tools and runtime engines.	Process design tools can export the graphical process descriptions in a standard format that other design tools can import without modification. An business process that relies only on web services can be executed on different runtime engines without modification.
The development effort for integration is reduced through the use of a BPM tool.	A business process can be described in a graphical fashion without requiring in-depth technical background knowledge about XML, SOAP, WSDL, etc. Process descriptions can be used by tools to create the stubs to begin the process implementations. The tools provide support for the integration of existing Web services into the business process.
Business processes can be changed dynamically at runtime.	New process versions can be deployed without causing downtime. Business rules can be changed on the fly. Tools support the deployment of new processes at runtime.

4 Designing and Implementing the Solution

4.1 DEFINING A SYSTEM ARCHITECTURE BASED ON THE T-CHECK CONTEXT

To design the solution, we created a notional architecture of the system based on the T-Check context discussed in Section 3.1. This architecture helped to determine the software requirements for the development and runtime environments. In compliance with the T-Check approach, the solution represents the simplest one possible that can be used to evaluate the hypotheses. Figure 5 illustrates the notional architecture designed for this T-Check investigation.

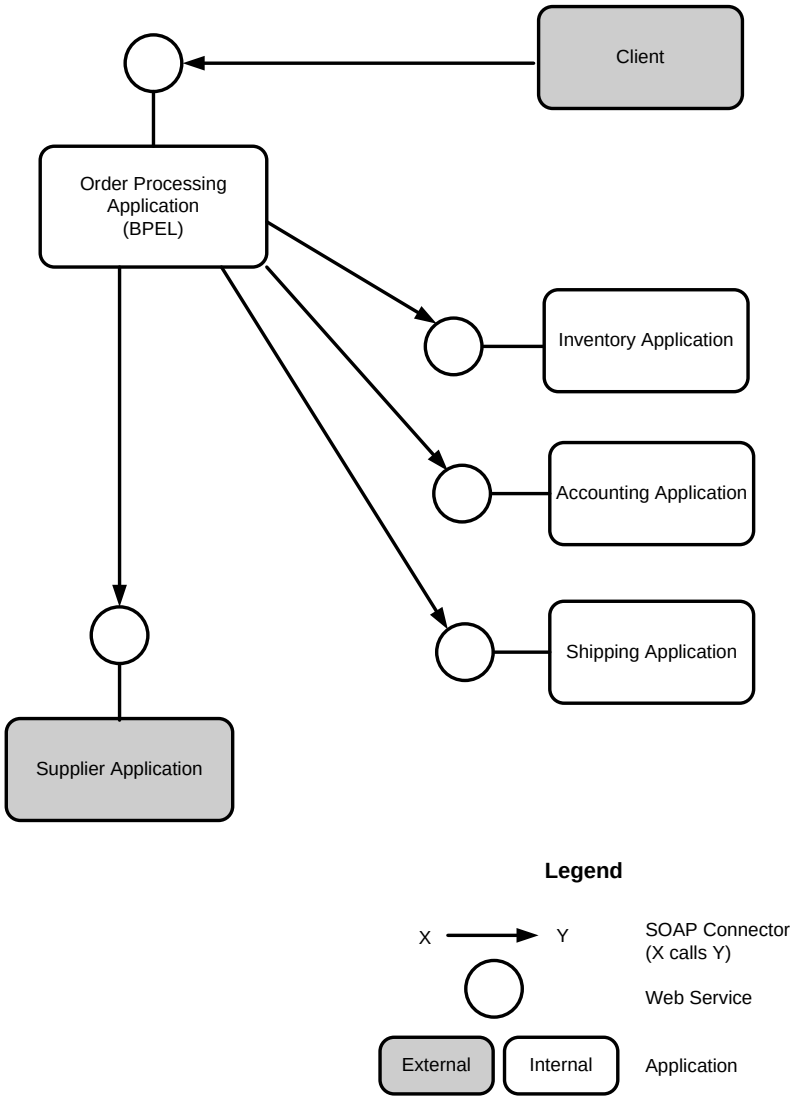

Figure 5: Notional System Architecture

The architecture contains the following components:

- The Client is a software client that sends SOAP requests to the Order Processing Application.

- The Order Processing Application accepts customer orders and processes them. The Order Processing Application is implemented as a BPEL process and executed on a BPEL runtime engine. The BPEL process exposes a Web service interface that the Client can use to submit orders.
- The Inventory Application is a Web service wrapper around the legacy inventory application. The Inventory Application checks the stock to decide whether enough goods are available to fulfill the order.
- The Accounting Application is a Web service wrapper around the legacy accounting application. This application looks up the prices for the ordered products and sends an invoice to the customer. The Accounting Application expects information about the order and the client in order to send out the invoice.
- The Shipping Application is another Web service wrapper around a legacy application. It is the link to the warehouse. When this application receives the shipment request, it initiates the process to ship the goods to the client.
- The Supplier Application represents the external product provider. This service is called when there are not enough items in stock to fulfill the client order.

For this T-Check investigation, we created implementations that provide rather trivial functionality for all components except the order processing application. This level of functionality is sufficient because the hypotheses are only concerned with the BPEL orchestration used to implement order processing. The functionality is as follows:

- The client is a simple .NET WinForms application to enter an order quantity. To save development time, we also use this application to simulate arrival of goods in inventory. This way we did not have to write a separate application to provide this functionality.
- Our inventory application returns the value *true* if the requested amount is less than six; otherwise, it returns the value *false*.
- Our accounting, shipping, and supplier applications simply log the incoming request messages.

Our testing scenario requires the use of four external systems. Although the hypotheses could be evaluated with a smaller number of orchestrated services, four seems to be a realistic number for this integration project. At the same time, using four allows a certain complexity in the experiment.

Figure 6 shows a flow chart of the implemented business process.[3] The steps it illustrates are as follows:

1. The order arrives.
2. On arrival of an order message, for every order item, check if the ordered quantity is available in local inventory.
    a. If the item is available (Yes), send item to the customer immediately.
    b. If the item is not available (No), order item from the supplier. Upon arrival of the item, send it to the customer.

[3] The process is not very realistic because it is inefficient, but it fulfills the needs of the evaluation.

3. After sending an item to the customer, check if all order items have been shipped.
   a. If all items have been shipped, bill the customer.
   b. If all items have not been shipped, wait for items to arrive.

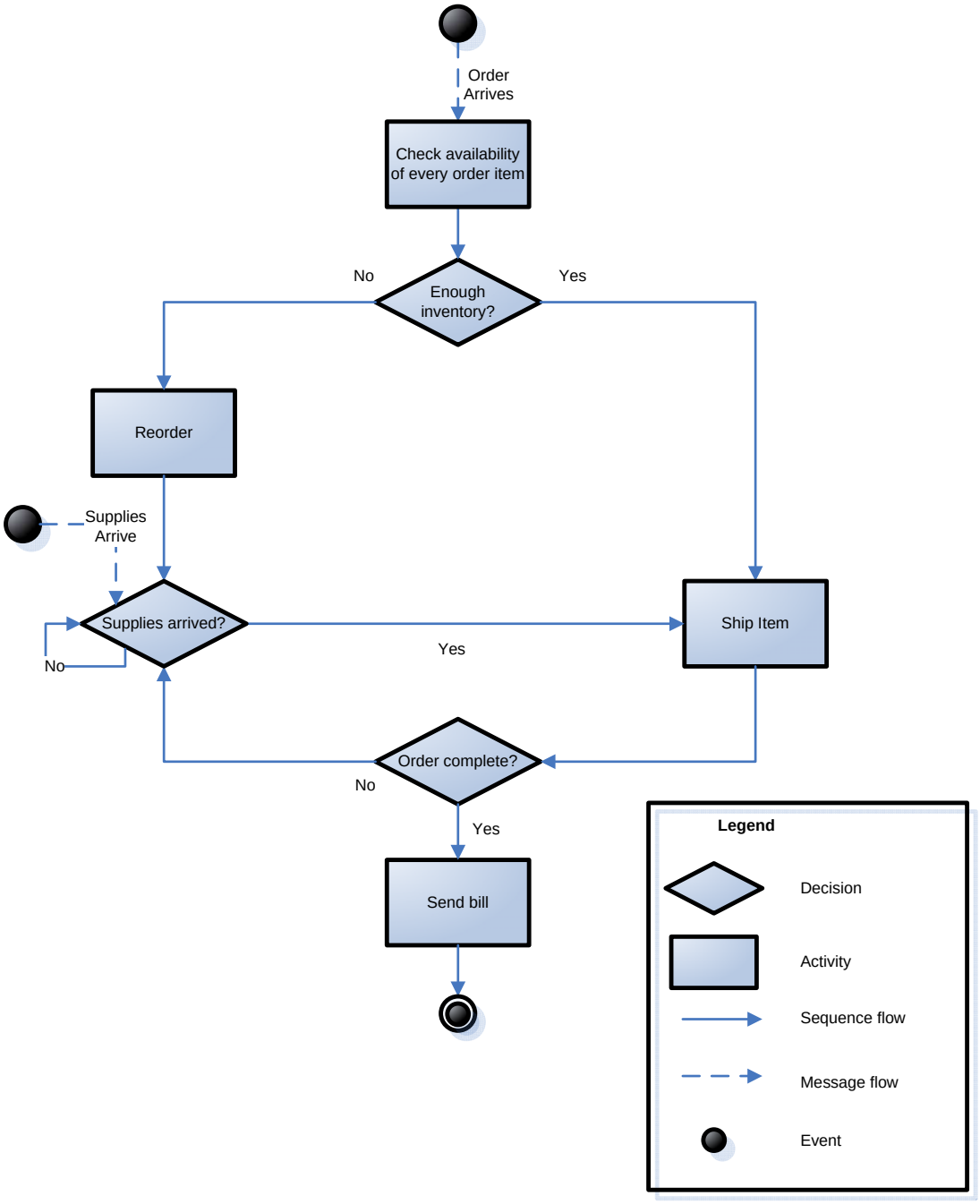

Figure 6: Flow Chart of the Order Processing Business Process

The mapping between the activities in Figure 6 and the elements of the notional architecture is shown in Figure 7.

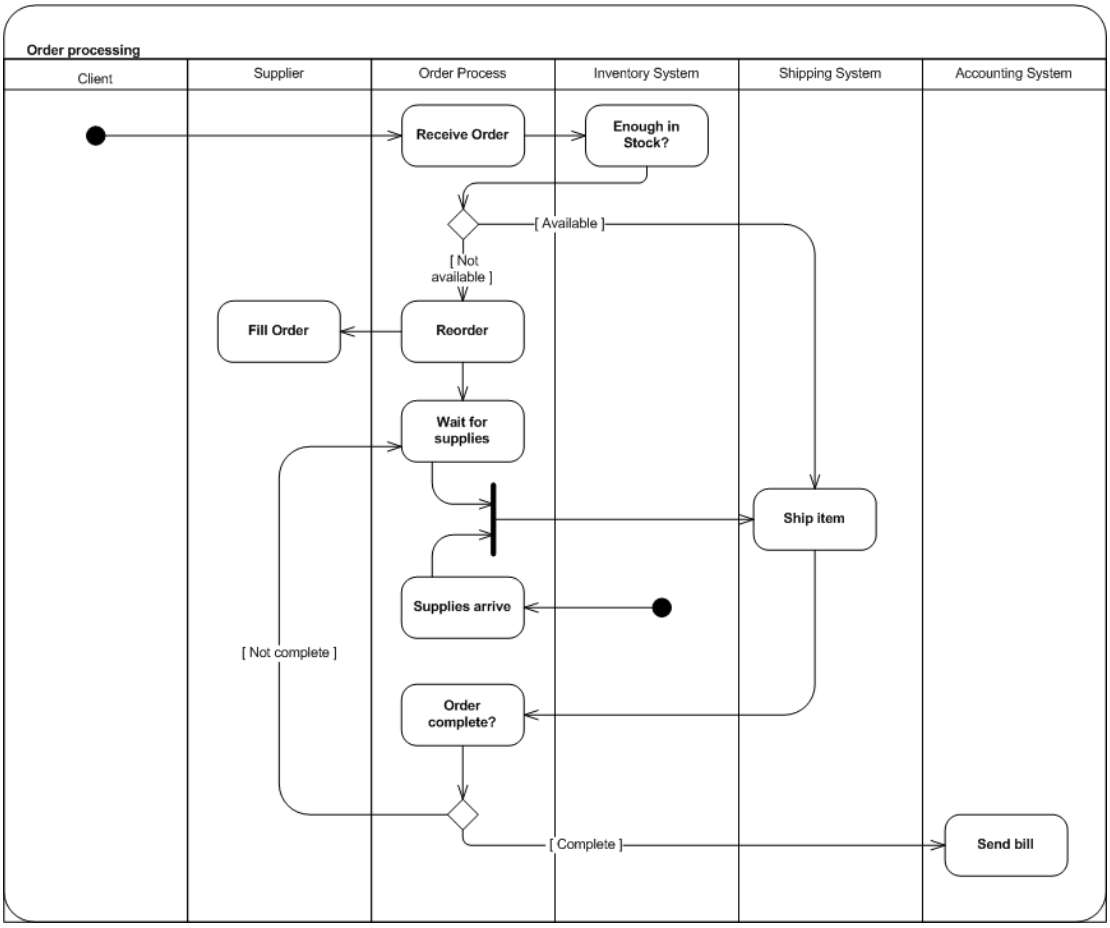

Figure 7: Order Processing Activities (UML Activity Diagram)

4.2 SELECTING TOOLS FOR DEVELOPMENT AND RUNTIME

One constraint on this T-Check investigation was a limited budget for the implementation. Therefore, we developed the solution using only tools that were available without charge. Another constraint was the availability of a stable release (at least a 1.0 version) of the products used. For the implementation of the web services, we used tools we were familiar with and that had a certain market acceptance. The tools we used are as follows:

- Apache Tomcat 6.0—a Java Servlet container to host the Web services and a simple web client application [Apache 2007]
- Apache Axis 2.0—a set of development tools and runtime libraries for web services development [Apache 2005]
- Microsoft IIS 6.0—a web server for the Microsoft Windows XP operating system that can host web services
- Sun Microsystems Application Server PE 9—a web and application server that hosts Java web services that also provides a BPEL runtime

To host the web services, we selected Tomcat/Axis and IIS because we had previous experience using them. Sun's Application Server was added as a web service host because we also wanted to

use it for testing the BPEL interoperability. Using it as web service host also allowed the simulation of the diversity of legacy systems which could be expected in the scenario.

For BPEL development, there was also limited budget; therefore, we selected two free products that had some good user reports:

- Active Endpoint ActiveBPEL Designer v4.1
- Sun Microsystems NetBeans IDE 6.0

It should be noted that we are not endorsing these tools and that we did not go through a rigorous selection process. We chose these tools because we are familiar with them and they are available at no cost. These considerations will likely be different in other projects.

4.3 IMPLEMENTING THE T-CHECK SOLUTION

We used Active Endpoints ActiveBPEL Designer to implement the orchestration. This product provides a graphical user interface that allows the creation of a BPEL process via drag-and-drop functionality.

To begin the orchestration process, the WSDL files of all the involved services need to be available. In the T-Check scenario, the legacy systems already have web service wrappers. The WSDL files from these services can be imported into the BPEL tool.

Because the BPEL process itself needs to be made available as a web service by the BPEL runtime, the next step is to define the service interface of the orchestration. We used a top-down approach, which means that we created the WSDL file manually in a WSDL editor. Alternatively, we could have used a bottom-up approach where the WSDL file is generated based on a pre-existing service implementation.

Having all the service interfaces in place, we defined the required basic and structured BPEL activities in the graphical ActiveBPEL editor. All messages must have an identifier, so that the BPEL engine knows which process to activate and which of the outstanding items is ready for shipment. This coordination between a waiting process and an arriving asynchronous message is called correlation in BPEL. Figure 8 shows a view of the ordering process as implemented. The document symbols with arrows represent <assign> activities. In this process, they are used to copy information between the messages. For example, in the AssignSupplyInfo activity, the number of items that need to be reordered is set in the message that is sent to the supplier. In our solution, this number is calculated as the difference between items ordered by the customer and items in stock. The scope shown within the <forEach> activity loop is equivalent to scope in traditional programming languages.

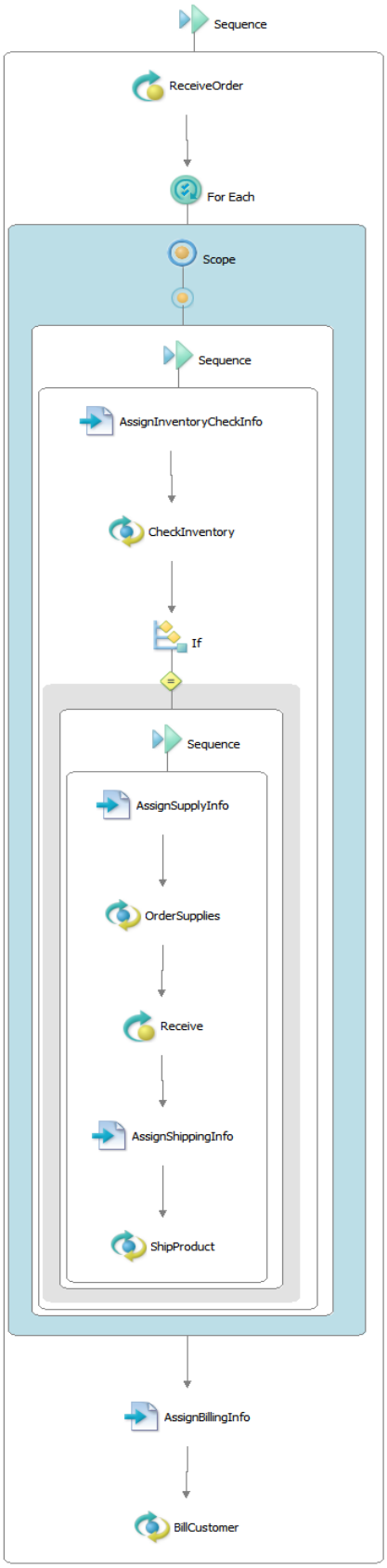

Figure 8: Ordering Process in ActiveBPEL

One important development task in building an orchestration is the specification of so called partner link types. A partner link type defines a relationship between the orchestration and a participating web services. These definitions are stored in a WSDL file. As an example, we show the definition of the partner link type for the inventory service:

```
<plnk2:partnerLinkType
       xmlns:plnk2="http://docs.oasis-open.org/wsbpel/2.0/plnktype"
       name="inventoryLT">
    <plnk2:role name="inventoryRole"
          portType="tns:InventoryServiceInterface"/>
</plnk2:partnerLinkType>
```

Each partner link type specifies at least one role. The role in the example is `inventoryRole`. The partner link type also names the port type that provides this role. The reference to `tns:InventoryServiceInterface` specifies a port type in the WSDL file of the inventory service.

One challenge in the process design is that, after a supply order is sent out, the process waits until the supplies arrive in the warehouse. Upon their arrival, a warehouse application recognizes that the delivered items are part of an outstanding customer order, and the Inventory Application sends a message to the Order Processing Application. Because modeling of this additional message did not add value to our hypothesis evaluation, we implemented this step through simple hard-coded reactivate messages sent from the client on demand.

All messages must have an identifier, so that the BPEL engine knows which process to activate and which of the outstanding items is ready for shipment. This coordination between a waiting process and an arriving asynchronous message is called correlation in BPEL. To make the order processing business process efficient, the inventory check, potential reordering, and waiting for supplies activities occur in parallel for all items. This flow means that there can be multiple parallel <receive> activities in the same process waiting for the same message from the same partner link. Overall, there is only one process instance per order, and each instance executes activities in parallel for each order item. Thus, it is not enough to use a correlation value unique to the process, but one that is unique to every reorder request. After the orchestration is complete, the correlation can be simulated with test values. The BPEL design tools support testing and debugging, because they provide support for connecting to a running process and inspecting its state and variable values.

The next step in the process design is to build a deployment descriptor for the orchestration. This descriptor is used by the BPEL runtime engine to find the needed WSDL files. Additionally, binding information can be specified. If, for example, the service endpoint location needs to be extracted dynamically from incoming messages, it can be defined here. In the last step, we package the BPEL process with the deployment descriptor and deploy it to the runtime engine. A copy of the deployment descriptor produced for this report is included in Appendix B. At this point the orchestration is ready for execution. Figure 9 shows how the parts of the system are deployed.

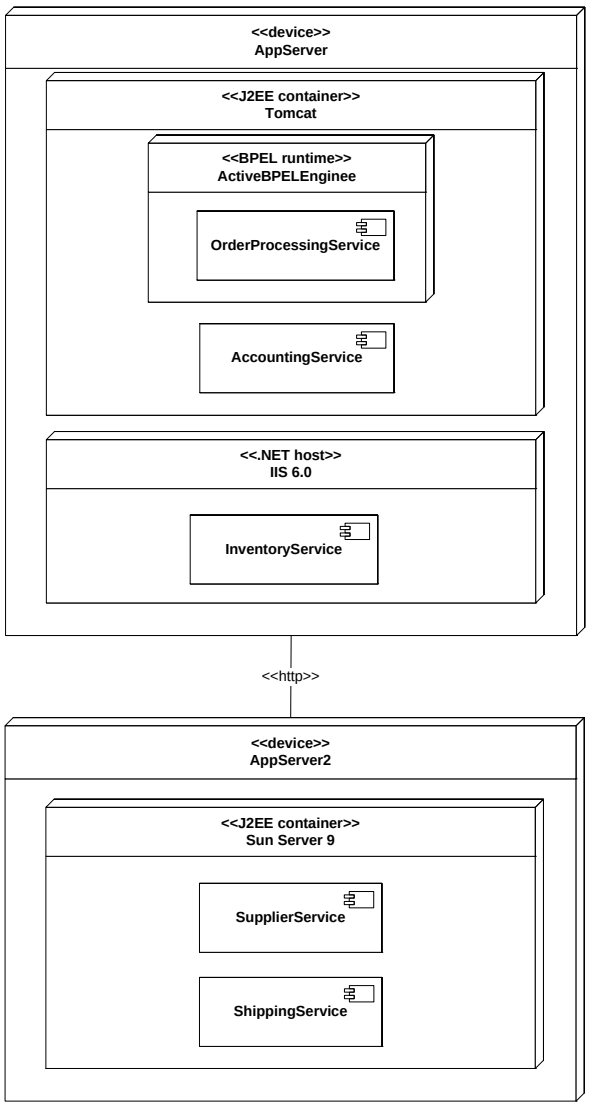

Figure 9: Deployment Diagram (UML)

4.4 IMPORTING THE BPEL PROCESS INTO NETBEANS

This section describes the adjustments required when the BPEL process was loaded in the second BPEL design tool, NetBeans 6.0. (Recall that our first hypothesis is that business process descriptions can be exchanged between different design tools and runtime engines.) First, we copied the process file into a new NetBeans "BPEL Module" project. The NetBeans BPEL editor laid out the process elements according to its rules. This behavior was expected, since BPEL does not save any graphical information about the process. Also there were a couple of error messages displayed:

- *Element "correlationSets" as child of "scope" is not supported by the Sun BPEL SE.*
  In the process, the correlation set is used to map order supply requests to the message that is sent from the Inventory Application, when supplies arrive. This correlation set was defined in the scope of the <forEach> loop. In the current version of the product, correlation sets are on-

ly allowed at the process level. We copied the declaration to the process element without problems.

- *Element "from" as child of "variable" is not supported by the Sun BPEL SE.*
  The <from> element was used in the process to initialize the process variable that holds the counter value while iterating over the order items in the client order. This element was easily replaced with an additional <assign> activity that set the counter to its initial value.

Additionally, there was a warning:

- *Attribute "parallel" is not yet supported by the Sun BPEL SE. This attribute will be ignored at runtime.*
  This warning applied to the <forEach> statement. The original process is designed so that it can iterate over all order items in parallel. This parallel activity includes checking the local stock and, if necessary, ordering and waiting for supplies and the shipment of the goods. Although the warning did not prevent the process from execution, it indicates that a bigger problem will occur during runtime. If the <forEach> branches cannot execute in parallel, the processing of the next order item can only start after the previous one has been shipped. If one of the order items has to be reordered from the supplier, the whole process is blocked until the supplies arrive. This consequence is especially unfortunate when all the subsequent order items could be shipped from the local stock. Of course, the problem could be avoided by redesigning the process to, for example, check the list of order items for availability in its own loop. After that check, the goods not in stock could be ordered while the available items could be shipped.

We resolved these and other issues by upgrading to version 9.1 of the Sun application server and making the following changes to the deployment descriptor:

1. We added binding information to all WSDL files, including the ones that describe the interface to the orchestration.
2. We updated the namespaces of some of the partner link type definitions. (These updates were done in response to warnings, so the process might have worked with the old namespaces.)
3. We converted the <query> expressions in the <copy> statements of the <assign> activities to ordinary XPath expressions. The runtime could not interpret the <query> statements. It took some experimentation to solve this problem because the NetBeans designer did not create any error or warning messages to indicate that the orchestration engine was not able to process the statements.

Overall, it required more effort than expected to make the orchestration work on another engine.

4.5 UPDATING A RUNNING BPEL PROCESS

To verify that it is possible to update a BPEL process that is currently executing, we created a variant of our main process that invokes shipping twice. In the tool, we started the execution of the original process and deployed the updated version during the execution of the original process. The tool continued execution of the already started process using the old process definition and used the updated process only for newly started processes.

5 Evaluation and Experience with the BPM Tools

In this section, we present the results of evaluating the solution against the criteria.

5.1 RESULTS FOR HYPOTHESIS 1

Hypothesis 1: **Business process descriptions can be exchanged between different design tools and runtime engines.**

This hypothesis is mostly sustained.

We encountered problems only in conjunction with (1) an incomplete implementation of the BPEL 2.0 specification and (2) deployment descriptors.

The first problem is a product-specific issue and thus not very interesting from the perspective of this evaluation. It is very likely that the next tool version will support the full standard. The problems encountered were simple and were fixed by making small adjustments to the process. Nonetheless, the issue is something that is good to know early in the development cycle.

The second problem is due to every tool in the experiment using its own version of the deployment descriptor. A deployment descriptor is used to link the BPEL processes with specifics about the runtime environment. For example, deployment descriptors contain the addresses of the partner services. There is no standard for the definition of these deployment descriptors, and therefore they need to be recreated when migrating between tools. The proprietary nature of the descriptors is not a surprise because at some point there needs to be a mapping from the process description to the functionality of the specific product. In bigger projects, it may be necessary to develop tool support for deployment descriptor conversion.

In addition to those problem areas, we noticed some areas where effort is required to use the BPEL tool:

- The graphical representation of the designed processes is not preserved when a BPEL process is opened with another BPEL design tool. This occurs because BPEL does not contain format or positioning information of elements within a process description. When a BPEL file is opened with a new tool, that product tries to lay out the process according to its own algorithms.
- The tools used for the implementation can also handle abstract processes. If a process is declared abstract, the tool's evaluation functionality issues errors if elements contained in the process are not allowed in an abstract processes. This functionality allows exporting an abstract process, which can then be used by a partner organization as a contract to develop the interaction with the BPEL process.

  This behavior becomes a problem only when product-specific extensions are used broadly for the implementation of a BPEL process. Both tools that we evaluated offered their own additional functionality. For example, ActiveBPEL supports access to some runtime properties such as ProcessId. Other tools from vendors such as Oracle and IBM offer ways to include workflows and business rules engines into BPEL processes. Also, there are extensions to directly access Java methods. The use of these extensions increases vendor lock-in. In

case of a migration to a different platform, that functionality would need to be re-implemented as extensions of the target orchestration engine.

In summary, there will most likely be some "tweaks" for the orchestration to work on a new platform, but the effort spent defining the process in one tool is not wasted when migrating to another tool. To maximize the reusability of processes, we recommend avoiding proprietary extensions where possible and documenting any use of extensions.

5.2 RESULTS FOR HYPOTHESIS 2

Hypothesis 2: **The development effort for integration is reduced through the use of a BPM tool.**

This hypothesis is not sustained in all cases.

It is important to clarify, though, that our analysis of effort reduction is not based on time because we have no data on the time required by other approaches for comparison. The criteria for this hypothesis look for tool functionality that would make the effort easier—specifically in graphical design, process implementation stubs, and integration with existing Web services.

There are two reasons we evaluate the tool as not fully sustaining our hypothesis. First, the effort for integration includes more than the development of the orchestration process itself. Because BPEL can only interact with Web services, there is additional effort required to develop participating services and wrappers around the legacy code that implements the processes. In some simple cases, it might even be easier to write a program in Java or some other language. Compared to Java, it is also cumbersome to express complex logic with the capabilities of BPEL and XPath expressions. As a result, there might be a need to develop additional services to encapsulate such functionality. These new services would be invoked from the BPEL process.

Second, the tools used in the T-Check could not completely hide the need for technical knowledge. Although the tools provide a graphical editor for creating an orchestration, the design elements are just a visual representation of the underlying technical concepts. Thus, certain knowledge of programming constructs (decisions, loops) is required. Tools that work at that level usually are fine for simple and medium complexity cases, but we doubt that they scale to more complex situations. A similar case is the use of graphical database query editors, where SQL knowledge will be required for writing complex queries. In BPEL, knowledge about WSDL or XPath expressions is necessary to define complex processes. Also, the definition of fault handlers is not easy to understand for a non-technical person. At the very least, technical knowledge will be required when the process is ready for deployment and possible errors need to be debugged.

There are tools in the market that work on a higher level of abstraction than BPEL. Products for business process definition, such as BPMN, allow a process to be drawn in a non-executable format. BPMN should be easy to understand for non-technical personnel such as business analysts. Some of these BPMN tools provide capabilities for generating BPEL code. But the creation of BPEL code from BPMN is a non-trivial task since BPMN and BPEL represent two different language classes [Ouyang 2006]. The experiments performed for this T-Check did not look into the BPEL code generation capabilities of BPMN tools. Such an investigation could be done as future work.

The tools we used in the experiments provided good support for the use of existing web services. They provided wizards that helped with the location of WSDL files, the selection of the required

ports and messages, and the creation of partner links. Therefore, development is simplified by the availability of these features that help with the integration of existing services.

The real benefit of having a process in BPEL is not a decrease of development time. It may be gained from the easy way in which a BPEL process can be changed and redeployed during its lifetime and the link to service repositories available to the people defining business processes. Overall, we do not have the impression that building a BPEL orchestration is always faster than implementing the same functionality as a traditional Java component. The choice to use BPEL or Java certainly depends on knowledge about them and their associated tools.

For the development effort, the BPEL capabilities used in the orchestration are most important. In that aspect, BPEL engines provide more support and Java may require more work, such as in the implementation of parallel activities and the correlation of processes. Also, the ability to specify a compensation process in case of a failure is a further benefit of BPEL. During compensation, the engine will restore all variables to the state they were in when the compensated scope was started. Additionally, runtime engines provide automatic dehydration capabilities. Dehydration describes the functionality of saving process state information for long-running processes to a file or database to save system resources. Additionally, state information needs to be saved in case of a system shutdown. On the arrival of the expected message, the engine will activate the dehydrated process and restore its state.

5.3 RESULTS FOR HYPOTHESIS 3

Hypothesis 3: **Business processes can be changed dynamically at runtime.**

This hypothesis is sustained.

The experiments have shown that one version of a process can be replaced with a newer version of that process without interrupting service execution. For the products we used, we observed that a running process was completed with the old process definition, and a new process was executed with the new definition. The question is whether this behavior is always desired from a business perspective.

The experiments performed did not look into the integration of business rules engines and BPEL engines, but we expect that a business rule engine would integrate well, as long as it provides a web services interface. In that case, the interaction between the two engines would work with standard BPEL activities (<invoke>, <receive>) and the capabilities for dynamic change of the rules would be dependent on the functionality of the business rules engine.

6 Future Work

There are two areas not covered in this T-Check investigation that could be of interest for organizations interested in BPM adoption. Those areas are

1. BPMN to BPEL mapping

   It would be interesting to see how well current tools create BPEL code from BPMN diagrams. Aspects to look at would be the amount of human effort that is required to make the code executable and the degree of support needed for round-trip engineering.

2. Workflow and business rule engine integration

   As mentioned in the report, some BPM products provide extensions for the incorporation of human tasks and business rules engines. Thus, a topic to look at further is how easily these extended processes can be changed at runtime.

7 Conclusions and Call for Response

Our T-Check investigation into BPM in a web services context shows that it is possible to exchange business process definitions between tools adhering to the standards in this area. Although the work for composing business processes is at a higher level than using traditional programming languages, it still requires a good level of technical knowledge. Even in the small scenario implemented in this T-Check, many different technologies and standards are involved and concepts such as correlation need to be understood. This is certainly not a level at which business analysts and other non-technical personnel can work. Overall, this T-Check investigation contributed to our understanding of how business processes can be composed out of independent web services.

The team in the Integration of Software-Intensive Systems (ISIS) Initiative at the SEI that is investigating BPM and other technologies using the T-Check approach is interested in feedback from and collaboration with the communities that are considering technologies for service-oriented environments. Write to the ISIS team at isis-sei@sei.cmu.edu.

Appendix A BPEL File for the Order Processing Application

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
BPEL Process Definition
Edited using ActiveBPEL(r) Designer Version 4.1.0 (http://www.active-endpoints.com)
-->
<bpel:process xmlns:abx="http://www.activebpel.org/bpel/extension"
      xmlns:bpel=http://docs.oasis-open.org/wsbpel/2.0/process/executable
      xmlns:ext="http://www.activebpel.org/2006/09/bpel/extension/query_handling"
      xmlns:ns="http://tempuri.org/InventoryMessages.xsd"
      xmlns:ns1="http://www.bpm-test.com/OrderingOrganization/"
      xmlns:ns2="http://testpartnerlink.com"
      xmlns:ns3="http://www.bpm-test.com/AccountingService/"
      xmlns:ns4="http://test_supplier.com/Supplier/"
      xmlns:ns5="http://www.bpm-test.com/ShipmentService/"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      ext:createTargetXPath="yes" name="OrderingProcess" suppressJoinFailure="yes"
      targetNamespace="http://OrderingProcess">

   <bpel:extensions>
      <bpel:extension mustUnderstand="yes"
            namespace="http://www.activebpel.org/2006/09/bpel/extension/query_handling"/>
   </bpel:extensions>

   <bpel:import importType="http://schemas.xmlsoap.org/wsdl/"
      location="InventoryService.wsdl"
      namespace="http://www.bpm-test.com/OrderingOrganization/"/>
   <bpel:import importType="http://schemas.xmlsoap.org/wsdl/"
      location=http://localhost/InventoryService/inventoryservice.asmx?WSDL
      namespace="http://www.bpm-test.com/OrderingOrganization/"/>
   <bpel:import importType="http://schemas.xmlsoap.org/wsdl/"
      location="AccountingService.wsdl"
      namespace="http://www.bpm-test.com/AccountingService/"/>
   <bpel:import importType="http://schemas.xmlsoap.org/wsdl/"
      location="ShipmentService.wsdl"
      namespace="http://www.bpm-test.com/ShipmentService/"/>
   <bpel:import importType=http://schemas.xmlsoap.org/wsdl/
      location="OrderingProcess.wsdl"
      namespace="http://www.bpm-test.com/OrderingOrganization/"/>
   <bpel:import importType="http://schemas.xmlsoap.org/wsdl/"
      location="SupplierService.wsdl"
      namespace="http://test_supplier.com/Supplier/"/>

   <bpel:partnerLinks>
      <bpel:partnerLink myRole="orderGoodsService" name="orderingLink"
            partnerLinkType="ns1:orderProcessLT"/>
      <bpel:partnerLink name="inventoryLT1" partnerLinkType="ns1:inventoryLT"
            partnerRole="inventoryRole"/>
      <bpel:partnerLink name="supplierLT1" partnerLinkType="ns4:supplierLT"
            partnerRole="supplierRole"/>
      <bpel:partnerLink name="accountingLT" partnerLinkType="ns3:accountingLT"
            partnerRole="AccountingRole"/>
      <bpel:partnerLink myRole="supplierCallBackRole" name="supplierCallBackLT"
            partnerLinkType="ns1:supplierCallBackLT"/>
      <bpel:partnerLink name="shippingLT" partnerLinkType="ns5:shippingLT" partnerRole="shippingRole"/>
   </bpel:partnerLinks>

   <bpel:messageExchanges><bpel:messageExchange name="Exchange1"/></bpel:messageExchanges>
```

```
<bpel:variables>
   <bpel:variable messageType="ns1:checkAvailabilityIn" name="checkAvailabilityIn"/>
   <bpel:variable messageType="ns1:checkAvailabilityOut" name="checkAvailabilityOut"/>
   <bpel:variable messageType="ns4:OrderSuppliesRequest" name="OrderSuppliesRequest"/>
   <bpel:variable messageType="ns4:OrderSuppliesResponse" name="OrderSuppliesResponse"/>
   <bpel:variable messageType="ns3:BillCustomerRequestMessage" name="BillCustomerRequestMessage"/>
   <bpel:variable messageType="ns3:BillCustomerResponseMessage" name="BillCustomerResponseMessage"/>
   <bpel:variable messageType="ns5:ShipProductRequest" name="ShipProductRequest"/>
   <bpel:variable messageType="ns5:ShipProductResponse" name="ShipProductResponse"/>
   <bpel:variable messageType="ns1:CustomerOrderMessage" name="CustomerOrderMessage"/>
   <bpel:variable messageType="ns1:OrderSuppliesCallbackRequest" name="OrderSuppliesCallbackRequest1"/>
   <bpel:variable name="SupplyOrdersCounter" type="xsd:int">
      <bpel:from>
         <bpel:literal>0</bpel:literal>
      </bpel:from>
   </bpel:variable>
</bpel:variables>

<bpel:sequence>
   <bpel:receive createInstance="yes" name="ReceiveOrder" operation="OrderProducts"
         partnerLink="orderingLink" portType="ns1:OrderProductsPT" variable="CustomerOrderMessage"/>
   <bpel:forEach counterName="counter" parallel="yes">
      <bpel:startCounterValue>1</bpel:startCounterValue>
      <bpel:finalCounterValue>count($CustomerOrderMessage.order/ns1:OrderItem)</bpel:finalCounterValue>
      <bpel:scope>
         <bpel:variables>
            <bpel:variable name="OrderItem" type="ns1:OrderItemType"/>
      </bpel:variables>
      <bpel:correlationSets>
         <bpel:correlationSet name="CS1" properties="ns4:SupplyCorrelationProperty"/>
      </bpel:correlationSets>
      <bpel:sequence>
         <bpel:assign name="AssignInventoryCheckInfo">
            <bpel:copy>
               <bpel:from part="order" variable="CustomerOrderMessage">
                  <bpel:query>ns1:OrderItem[$counter]</bpel:query>
               </bpel:from>
               <bpel:to variable="OrderItem"/>
            </bpel:copy>
            <bpel:copy>
               <bpel:from variable="OrderItem">
                  <bpel:query>ns1:productId</bpel:query>
               </bpel:from>
               <bpel:to part="messagePart" variable="checkAvailabilityIn">
                  <bpel:query>ns:productId</bpel:query>
               </bpel:to>
            </bpel:copy>
         </bpel:assign>
         <bpel:invoke inputVariable="checkAvailabilityIn" name="CheckInventory" operation="checkAvailability"
               outputVariable="checkAvailabilityOut" partnerLink="inventoryLT1"
                portType="ns1:InventoryServiceInterface"/>
            <bpel:if>
               <bpel:condition>
                  $checkAvailabilityOut.messagePart/ns:availableItems &lt; $OrderItem/ns1:amount
               </bpel:condition>
               <bpel:sequence>
                  <bpel:assign name="AssignSupplyInfo">
                     <bpel:copy>
                        <bpel:from variable="OrderItem">
                           <bpel:query>ns1:productId</bpel:query>
                        </bpel:from>
                        <bpel:to part="order" variable="OrderSuppliesRequest">
                           <bpel:query>ns4:OrderItem/ns4:productId</bpel:query>
```

```
            </bpel:to>
        </bpel:copy>
        <bpel:copy>
            <bpel:from>
                $OrderItem/ns1:amount - $checkAvailabilityOut.messagePart/ns:availableItems
            </bpel:from>
            <bpel:to part="order" variable="OrderSuppliesRequest">
                <bpel:query>ns4:OrderItem/ns4:amount</bpel:query>
            </bpel:to>
        </bpel:copy>
        <bpel:copy>
            <bpel:from>
                <bpel:literal>CustomerId19</bpel:literal>
            </bpel:from>
            <bpel:to part="order" variable="OrderSuppliesRequest">
                <bpel:query>ns4:customerId</bpel:query>
            </bpel:to>
        </bpel:copy>
        <bpel:copy>
            <bpel:from variable="SupplyOrdersCounter">
                <bpel:query>$SupplyOrdersCounter</bpel:query>
            </bpel:from>
            <bpel:to part="order" variable="OrderSuppliesRequest">
                <bpel:query>ns4:orderId</bpel:query>
            </bpel:to>
        </bpel:copy>
        <bpel:copy>
            <bpel:from>
                <bpel:literal>0</bpel:literal>
            </bpel:from>
            <bpel:to part="order" variable="OrderSuppliesRequest">
                <bpel:query>ns4:orderId</bpel:query>
            </bpel:to>
        </bpel:copy>
        <bpel:copy>
            <bpel:from variable="counter">
                <bpel:query>$counter</bpel:query>
            </bpel:from>
            <bpel:to part="order" variable="OrderSuppliesRequest">
                <bpel:query>ns4:orderId</bpel:query>
            </bpel:to>
        </bpel:copy>
    </bpel:assign>
    <bpel:invoke inputVariable="OrderSuppliesRequest" name="OrderSupplies"
            operation="OrderSupplies" outputVariable="OrderSuppliesResponse"
            partnerLink="supplierLT1" portType="ns4:Supplier">
        <bpel:correlations>
            <bpel:correlation initiate="yes" pattern="request" set="CS1"/>
        </bpel:correlations>
    </bpel:invoke>
    <bpel:receive operation="SupplierCallBack" partnerLink="supplierCallBackLT"
            portType="ns1:SupplierCallBackPT" variable="OrderSuppliesCallbackRequest1">
        <bpel:correlations>
            <bpel:correlation initiate="no" set="CS1"/>
        </bpel:correlations>
    </bpel:receive>
    <bpel:assign name="AssignShippingInfo">
        <bpel:copy>
            <bpel:from part="order" variable="CustomerOrderMessage">
                <bpel:query>ns1:CustomerInfo/ns1:Name</bpel:query>
            </bpel:from>
            <bpel:to part="ShipProductRequest" variable="ShipProductRequest">
                <bpel:query>Name</bpel:query>
```

```
                    </bpel:to>
                </bpel:copy>
                <bpel:copy>
                    <bpel:from part="order" variable="CustomerOrderMessage">
                        <bpel:query>ns1:CustomerInfo/ns1:Address</bpel:query>
                    </bpel:from>
                    <bpel:to part="ShipProductRequest" variable="ShipProductRequest">
                        <bpel:query>Address</bpel:query>
                    </bpel:to>
                </bpel:copy>
                <bpel:copy>
                    <bpel:from part="order" variable="CustomerOrderMessage">
                        <bpel:query>ns1:CustomerInfo/ns1:City</bpel:query>
                    </bpel:from>
                    <bpel:to part="ShipProductRequest" variable="ShipProductRequest">
                        <bpel:query>City</bpel:query>
                    </bpel:to>
                </bpel:copy>
                <bpel:copy>
                    <bpel:from variable="OrderItem">
                        <bpel:query>ns1:productId</bpel:query>
                    </bpel:from>
                    <bpel:to part="ShipProductRequest" variable="ShipProductRequest">
                        <bpel:query>ProductId</bpel:query>
                    </bpel:to>
                </bpel:copy>
                <bpel:copy>
                    <bpel:from variable="OrderItem">
                        <bpel:query>ns1:amount</bpel:query>
                    </bpel:from>
                    <bpel:to part="ShipProductRequest" variable="ShipProductRequest">
                        <bpel:query>Amount</bpel:query>
                    </bpel:to>
                </bpel:copy>
            </bpel:assign>
            <bpel:invoke inputVariable="ShipProductRequest" name="ShipProduct" operation="ShipProduct"
                outputVariable="ShipProductResponse" partnerLink="shippingLT"
                portType="ns5:ShipmentService"/>
        </bpel:sequence>
    </bpel:if>
</bpel:sequence>
</bpel:scope>
</bpel:forEach>
<bpel:assign name="AssignBillingInfo">
    <bpel:copy>
        <bpel:from>
            <bpel:literal>OrderId555</bpel:literal>
        </bpel:from>
        <bpel:to part="BillCustomer" variable="BillCustomerRequestMessage">
            <bpel:query>OrderId</bpel:query>
        </bpel:to>
    </bpel:copy>
    <bpel:copy>
        <bpel:from>
            <bpel:literal>
                <BillingPositionType xmlns:ns5="http://http://www.bpm-test.com/ShipmentService/"
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="ns3:BillingPositionType">
                    <Description>string</Description>
                    <Price>10</Price>
                </BillingPositionType>
            </bpel:literal>
        </bpel:from>
        <bpel:to part="BillCustomer" variable="BillCustomerRequestMessage">
```

```
                <bpel:query>BillingPosition</bpel:query>
            </bpel:to>
        </bpel:copy>
        <bpel:copy>
            <bpel:from>
                <bpel:literal>100</bpel:literal>
            </bpel:from>
            <bpel:to part="BillCustomer" variable="BillCustomerRequestMessage">
                <bpel:query>Total</bpel:query>
            </bpel:to>
        </bpel:copy>
        <bpel:copy>
            <bpel:from part="order" variable="CustomerOrderMessage">
                <bpel:query>ns1:CustomerInfo/ns1:Name</bpel:query>
            </bpel:from>
            <bpel:to part="BillCustomer" variable="BillCustomerRequestMessage">
                <bpel:query>CustomerAddress/Name</bpel:query>
            </bpel:to>
        </bpel:copy>
        <bpel:copy>
            <bpel:from part="order" variable="CustomerOrderMessage">
                <bpel:query>ns1:CustomerInfo/ns1:Address</bpel:query>
            </bpel:from>
            <bpel:to part="BillCustomer" variable="BillCustomerRequestMessage">
                <bpel:query>CustomerAddress/Street</bpel:query>
            </bpel:to>
        </bpel:copy>
        <bpel:copy>
            <bpel:from part="order" variable="CustomerOrderMessage">
                <bpel:query>ns1:CustomerInfo/ns1:City</bpel:query>
            </bpel:from>
            <bpel:to part="BillCustomer" variable="BillCustomerRequestMessage">
                <bpel:query>CustomerAddress/City</bpel:query>
            </bpel:to>
        </bpel:copy>
    </bpel:assign>
    <bpel:invoke inputVariable="BillCustomerRequestMessage" name="BillCustomer" operation="BillCustomer"
        outputVariable="BillCustomerResponseMessage" partnerLink="accountingLT"
        portType="ns3:AccountingService"/>
  </bpel:sequence>
</bpel:process>
```

Appendix B ActiveBPEL Deployment Descriptor

```
<?xml version="1.0" encoding="UTF-8"?>
<process xmlns="http://schemas.active-endpoints.com/pdd/2006/08/pdd.xsd"
      xmlns:bpelns="http://OrderingProcess"
      xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing"
      location="bpel/OrderingOrganization/OrderingProcess.bpel" name="bpelns:OrderingProcess">
   <partnerLinks>
      <partnerLink name="accountingLT">
         <partnerRole endpointReference="static">
            <wsa:EndpointReference xmlns:s="http://www.bpm-test.com/AccountingService/"
                  xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing">
               <wsa:Address>http://localhost:6502/AccountingService/services/AccountingService</wsa:Address>
               <wsa:ServiceName PortName="AccountingServiceSOAP">s:AccountingService</wsa:ServiceName>
            </wsa:EndpointReference>
         </partnerRole>
      </partnerLink>
      <partnerLink name="inventoryLT1">
         <partnerRole endpointReference="static">
            <wsa:EndpointReference xmlns:s="http://www.bpm-test.com/OrderingOrganization/"
                  xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing">
               <wsa:Address>http://localhost/InventoryService/inventoryservice.asmx</wsa:Address>
               <wsa:ServiceName PortName="InventoryServiceBinding">s:InventoryServiceBinding</wsa:ServiceName>
            </wsa:EndpointReference>
         </partnerRole>
      </partnerLink>
      <partnerLink name="orderingLink">
         <myRole allowedRoles="" binding="RPC-LIT" service="orderingLinkService"/>
      </partnerLink>
      <partnerLink name="shippingLT">
         <partnerRole endpointReference="static">
            <wsa:EndpointReference xmlns:s="http://www.bpm-test.com/ShipmentService/"
                  xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing">
               <wsa:Address>http://appserver2:8090/Shipment/ShipmentService</wsa:Address>
               <wsa:ServiceName PortName="ShipmentServiceSOAP">s:ShipmentService</wsa:ServiceName>
            </wsa:EndpointReference>
         </partnerRole>
      </partnerLink>
      <partnerLink name="supplierCallBackLT">
         <myRole allowedRoles="" binding="RPC-LIT" service="supplierCallBackLTService"/>
      </partnerLink>
      <partnerLink name="supplierLT1">
         <partnerRole endpointReference="static">
            <wsa:EndpointReference xmlns:s="http://test_supplier.com/Supplier/"
                  xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing">
               <wsa:Address>http://appserver2:8090/Supplier/SupplierService</wsa:Address>
               <wsa:ServiceName PortName="SupplierPort">s:SupplierService</wsa:ServiceName>
            </wsa:EndpointReference>
         </partnerRole>
      </partnerLink>
   </partnerLinks>
   <references>
      <wsdl location="file:/SupplierService.wsdl" namespace="http://test_supplier.com/Supplier/"/>
      <wsdl location="file:/InventoryService.wsdl" namespace="http://www.bpm-test.com/OrderingOrganization/"/>
      <wsdl location="file:/ShipmentService.wsdl" namespace="http://www.bpm-test.com/ShipmentService/"/>
      <wsdl location="file:/OrderingProcess.wsdl" namespace="http://www.bpm-test.com/OrderingOrganization/"/>
      <wsdl location="file:/AccountingService.wsdl" namespace="http://www.bpm-test.com/AccountingService/"/>
   </references>
</process>
```

References

URLs are valid as of the publication date of this document.

[Apache 2007]
The Apache Software Foundation. *Apache Tomcat*. http://tomcat.apache.org/ (1999−2007)

[Eclipse 2007]
Eclipse. *Eclipse – an open development platform*. http://www.eclipse.org/ (2007)

[IBM 2007]
IBM, SAP. *Web WS-BPEL Extension for People (BPEL4People), Version 1.0*
http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-bpel4people/BPEL4People_v1.pdf (2007)

[Intalio 2008]
Intalio. *Downloads*. http://bpms.intalio.com/downloads.html (1999–2008)

[Leymann 2006]
Leymann, F., Roller, D., Schmidt, M.-T. "Web services and business process management." *IBM Systems Journal 41*, 2 (2002): 199–211.
http://www.research.ibm.com/journal/sj/412/leymann.pdf (2002)

[Lewis 2005]
Lewis, Grace A. & Wrage, Lutz. *A Process for Context-Based Technology Evaluation* (CMU/SEI-2005-TN-025, ADA441251). Software Engineering Institute, Carnegie Mellon University, 2005. http://www.sei.cmu.edu/publications/documents/06.reports/05tn025.html

[Lewis 2006]
Lewis, Grace A. & Wrage, Lutz. *Model Problems in Technologies for Interoperability: Web Services* (CMU/SEI-2006-TN-021, ADA454363). Software Engineering Institute, Carnegie Mellon University, 2006. http://www.sei.cmu.edu/publications/documents/06.reports/06tn021.html

[OASIS 2005]
Organization for the Advancement of Structured Information Standards. *OASIS UDDI*.
http://www.uddi.org/ (2005)

[OASIS 2007a]
Organization for the Advancement of Structured Information Standards. *Web Services Coordination (WS-Coordination) Version 1.1.*
http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.1-spec/wstx-wscoor-1.1-spec.html (2007)

[OASIS 2007b]
Organization for the Advancement of Structured Information Standards. *Web Services Atomic Transaction (WS-AtomicTransaction) Version 1.1* http://docs.oasis-open.org/ws-tx/wstx-wsat-1.1-spec/wstx-wsat-1.1-spec.html (2007)

[OASIS 2007c]
Organization for the Advancement of Structured Information Standards. *Web Services Business Activity (WS-BusinessActivity) Version 1.1* http://docs.oasis-open.org/ws-tx/wstx-wsba-1.1-spec-errata-os/wstx-wsba-1.1-spec-errata-os.html (2007)

[OASIS 2007d]
Organization for the Advancement of Structured Information Standards. *Web Services Business Process Execution Language Version 2.0*. http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html/ (2007)

[OMG 2006]
Object Management Group. *Unified Modeling Language (UML), version 2.1.1* http://www.omg.org/technology/documents/formal/uml.htm (2006)

[OMG 2007]
Object Management Group. *Business Process Modeling Notation (BPMN) Specification.* http://www.omg.org/cgi-bin/apps/doc?dtc/07-06-03.pdf (2007)

[Ouyang 2006]
Ouyang, Chun, Dumas, Marlon, ter Hofstede, Arthur H. M., & van der Aalst, Wil M. P. "From BPMN Process Models to BPEL Web Services," 285–292. *International Conference on Web Services (ICWS 2006)*. Salt Lake City, UT (USA), September 2006. IEEE Computer Society, 2006.

[SearchSOA 2005]
SearchSOA.com. *Dancing with Web services: W3C chair talks choreography*. http://searchsoa.techtarget.com/qna/0,289202,sid26_gci1066118,00.html (2005)

[W3C 1999]
World Wide Web Consortium. *XML Path Language Version 1.0*. http://www.w3.org/TR/xpath (1999)

[W3C 2003]
World Wide Web Consortium. *HTTP - Hypertext Transfer Protocol*. http://www.w3.org/Protocols/ (2003)

[W3C 2004]
World Wide Web Consortium. *Web Services Architecture*. http://www.w3.org/TR/ws-arch/ (2004)

[W3C 2005a]
World Wide Web Consortium. *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. W3C Working Draft 3 August 2005*. http://www.w3.org/TR/wsdl20/ (2005)

[W3C 2005b]
World Wide Web Consortium. *Description of W3C Technology Stack Illustration*. http://www.w3.org/Consortium/techstack-desc.html (2005)

[W3C 2005c]
World Wide Web Consortium. *Web Services Web Services Choreography Description Language Version 1.0. W3C Candidate Recommendation 9 November 2005*. http://www.w3.org/TR/ws-cdl-10/ (2005)

[WfMC 2005]
Workflow Management Coalition. *Process Definition Interface—XML Process Definition Language*. http://www.wfmc.org/standards/docs.htm#XPDL_Spec_Final (2005)

[Wikimedia 2007]
Wikimedia Foundation. *Business Process Management*. http://en.wikipedia.org/wiki/Business_Process_Management (2007)

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	September 2008	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
T-Check in Technologies for Interoperability: Business Process Management in a Web Services Context	FA8721-05-C-0003

6. AUTHOR(S)
Fabian Hueppi, Lutz Wrage, Grace A. Lewis

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2008-TN-005

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116	

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

In Business Process Management (BPM), many technologies are available to describe, analyze, execute, and monitor business processes. Composition languages are one type of BPM technology. Through the use of composition languages, business processes that are implemented through software and available as web services can be combined into new processes. The most popular language in this field is the Business Process Execution Language (BPEL). BPEL allows a user to declaratively combine existing services within and outside an organization to implement a full business process. This technical note presents the results of applying the T-Check approach in an initial investigation of BPEL and related technologies for the implementation of BPM. This approach involves (1) formulating hypotheses about the technology and (2) examining these hypotheses against specific criteria through hands-on experimentation. The outcome of this two-stage approach is that the hypotheses are either fully or partially sustained or refuted. In this report, three hypotheses are examined: (1) business process descriptions can be exchanged between different design tools and runtime engines; (2) the development effort for integration is reduced through the use of a BPM tool; and (3) business processes can be changed dynamically at runtime. From the T-Check investigation, the first two hypotheses are partially sustained and the last hypothesis is fully sustained.

14. SUBJECT TERMS	15. NUMBER OF PAGES
T-Check, Tcheck, business process management, BPM, BPEL, interoperability	44

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL